(12)

AD-A163 113

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 85 - 07 - 04	2. GOVT ACCESSION NO. ADA 163113	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Quarter Horse: A Case Study in Rapid Prototyping of a 32-bit Microprocessor Chip		5. TYPE OF REPORT & PERIOD COVERED Technical, interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) S. Ho, B. Jinks, T. Knight, J. Schaad, L. Snyder, A. Tyagi, C. Yang		8. CONTRACT OR GRANT NUMBER(s) MDA903-85-K-0072 ARPA-4563, #2, Code 5D30. ARPA Order-4563
9. PERFORMING ORGANIZATION NAME AND ADDRESS UW/NW VLSI Consortium, Dept. of Computer Science FR-35, University of Washington, Seattle, 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA - IPTO 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE July, 1985
		13. NUMBER OF PAGES 13
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ONR University of Washington 315 University District Building 1107 NE 45th St., JD-16, Seattle, WA 98195		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE JAN 1 4 1986 S D E

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

VLSI, RISC, MIPS, PLA, RAM, ALU, CMOS, MOSIS, SPICE choice complexity, & (end)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Quarter Horse is a single chip 32-bit microprocessor whose design and implementation in custom CMOS was completed in 90 days. The design effort is presented as a case study in managing choice complexity. The factors contributing to the rapid development of a prototype are discussed, as is the processor's architecture. Keywords include:

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

86 1 14 037

DTIC FILE COPY

The Quarter Horse: A Case Study in Rapid Prototyping of a 32-bit Microprocessor Chip

S. Ho, B. Jinks, T. Knight, J. Schaad,
L. Snyder, A. Tyagi, C. Yang

Department of Computer Science
Seattle, Washington 98195

Technical Report 85-07-04
July, 1985

ABSTRACT

The Quarter Horse is a single chip 32-bit microprocessor whose design and implementation in custom CMOS was completed in 90 days. The design effort is presented as a case study in managing *choice complexity*. The factors contributing to the rapid development of a prototype are discussed, as is the processor's architecture.

Funded in part by the Defense Advanced Research Projects Agency under Contract MDA903-85-K-0072, and in part by the University of Washington / Northwest VLSI Consortium. This paper appeared in the 1985 International Conference on Computer Design.

Our minds are finite, and yet even in these circumstances of finitude we are surrounded by possibilities that are infinite.
— Alfred North Whitehead

We must never make experiments to confirm our ideas, but simply to control them.
— Claude Bernard

1 Introduction

The difficulties of implementing large VLSI designs derive from two quite different sources which can be named: mass complexity and choice complexity. Mass complexity refers to the difficulty of specifying the detail in a large VLSI design *when it is known what is to be done*; this type of complexity is primarily a consequence of the sheer quantity of information. (Examples will be given shortly.) Choice complexity refers to the difficulty of selecting among alternatives within the design space *to determine how it is to be done*; this type of complexity is due to the complicated interactions of the components and the interdependence of design decisions. To complete a design is to triumph over mass complexity; to complete a *successful* design is to triumph over both types of complexity.

There are two purposes in distinguishing between mass and choice complexity. First, the distinction helps to clarify what design problems can and can not be solved by CAD tools and VLSI design methodologies. For example, the hierarchical design methodology is directed purely at managing mass complexity: The hundreds of thousands of polygons that make up a layout are organized into a few composite cells and leaf cells. Design rule and other checking facilities also aid in handling mass complexity since most designers can produce correct primitive cells but introduce design rule errors in cell composition or design revision. Simulation tools, on the otherhand, help confront choice complexity by giving designers the ability to evaluate design alternatives in software rather than hardware. Methodologies such as NORA (No Race logic [1]) constrain the design and simplify dynamic behavior, thereby reducing choice complexity. Other tools and methodologies can be similarly classified and it is interesting to assess how they carve away at the overall complexity problem.

The second reason to separate mass from choice complexity is to study ways of handling choice complexity, the more difficult and less well understood of the two. The problem is that simulators, the principal computer aid for coping with choice complexity, quickly become inadequate. The computational load of simulation becomes too large to be feasible; the simple, approximate models that work well on small designs fail to be accurate when scaled up; the space of possibilities becomes too large to be explored fully as the number of interacting and interdependent parts increases. The difficulties of choice complexity increase nonlinearly with the size of the design, causing it to be the more significant limitation.

Clearly, the easiest way to handle choice complexity is through *experience*, since making design decisions is largely a matter of "knowing from experience" the consequences of choosing various alternatives. But it is not possible to start a design and make every decision correctly from experience because were it so, every decision would have been previously made and the design would be essentially a copy of a previous effort — no *design* would be required. So, design decisions must be made when there is no direct experience to guide the designer. He must therefore find ways of acquiring the relevant experience. This paper focuses on the acquisition and application of experience — the management of choice complexity.

The Quarter Horse, a 32-bit microprocessor chip, designed start-to-finish in 90 days, will serve as a case study for discussing choice complexity management. The rapid 90-day design time is itself a means of quickly acquiring experience, but there are many other more fundamental instances within the design where choice complexity was reduced by bringing our experience or other people's experience to bear on the problem. Most, but not all, of these strategies are applicable to other large VLSI designs.

The organization of the paper is to give an architectural description of the Quarter Horse chip together with an inline discussion of choice complexity. In the Summary section, we will extract some general principles that can be applied to other designs.

2 The Quarter Horse Architecture

In this section we present the Quarter Horse architecture. Concurrently, we identify instances of choice complexity and describe ways that it may be reduced.

2.1 Project Objectives

Perhaps the most significant way in which choice complexity was controlled in the Quarter Horse chip was by selecting an achievable, intermediate goal. Rather than building a high performance microprocessor as our first objective, we chose a straightforward, unadorned machine *which was intended to be redesigned.* This approach, which exploits the tendency of second implementations to be cleaner, more efficient, more stable, etc., provided many opportunities for simplification.

First, decisions did not *have* to be right the first time, since they could be changed later. We could forge ahead on the likelihood, rather than the certainty, that a decision was correct and thus save exploration time. Second, by ignoring in the first design the features that would improve the performance of the processor (pipelining, caching, or whatever), we saved work in the short run, thus enabling the chip to be completed sooner, and giving us quicker feedback on our decisions. Third, solutions to certain problems could be delayed until the information needed to solve them was available. Global floor planning is such a problem. Until they are stable and the inter-relationships are fixed, there is no point in packing the components together tightly. We placed the components with generous space separating them in order to provide for last minute changes in size; see Figure 1. To do this within the confines of our die size we used the real estate that would have been assigned to the unimplemented performance features.

Therefore, the intermediate goal of the Quarter Horse effort was to build a basic 32-bit microprocessor that could be enhanced to be a high performance machine in a follow-on design. Since it was impossible to predict how the follow-on design would affect the Quarter Horse, the processor could be self contained if it had flexibility and simplicity. We were designing for experience, but if parts of the design were good enough to keep in later designs, that would be an added benefit.

2.2 The Architecture

There is a long (by the standards of the IC world) tradition of single chip microprocessors, so one way to reduce choice complexity is to learn from other project's successes and failures. The resulting architecture (see Figure 2) benefitted greatly from this experience. (Explicit credit is given below.)

The processor is built around two 32-bit wide data path busses with a control PLA and a one-instruction prefetch unit. The data path is connected at one end to a 32-bit address port and at the other end to a 32-bit data port.

It is interesting to note that although we advocate aggressive exploration of the design space using every means possible, there are a few choices that just cannot be made easily. These decisions tend to be global and one example is the use of the chip's pins: There is the address port/data port (ap/dp) choice versus the instruction port/value port (ip/vp) choice. In the ap/dp choice, all addresses go through the same pins while the data port must be shared between instructions and values. In the ip/vp choice the pins must be shared between addresses and data. The consequences of the decision affect the internals of the chip significantly and the interface of the microprocessor with the memory system. Although the ip/vp choice was never selected in the microprocessors we studied, our evaluation was that both schemes had assets and liabilities. Our final decision was to choose ap/dp more or less arbitrarily; perhaps it would be worth another 90 days to explore fully the ip/vp choice.

The Quarter Horse has a limited number of instructions like the RISC architecture [2] although we allocated a full 32 bits for the instruction format as was used in the PP4 [3] in order to have flexibility for later expansion. The instruction format is:

```
<----------------------- 32 ------------------------------>
|op code |SCC|IMM|  D   |  S1   |  S2   |          Immediate |
    8      1   1    5      5       5
                                 <-------------12------------->
```

where the abbreviations are:

SCC	Set condition code
IMM	Immediate data flag
D	Result destination register
S1	Source data register 1
S2	Source data register 2

When immediate values are used S2 is not available.

The architecture uses the register-to-register approach as was used in RISC. Unlike RISC, however, instructions employ a variable number of microcycles which are specified by the controlling PLA. (The purpose and benefits of this choice are explained below.) As a result, instructions can be implemented whose complexity is too great to be completed in a rigid, fixed length fetch/execute cycle. We chose to illustrate this property with instructions using character type data.

The Quarter Horse employs word addressing, as was used in MIPS [4]. Furthermore, we limited the memory space to 2^{28} words, which is quite adequate for our experimental purposes and enables interrupt addresses to be stored in 28 bits. This modest limitation and the fact that the four flag bits fit in the unused positions permit a one word program status word.

2.3 Architectural Components

We briefly discuss each of the components of the architecture with a concurrent discussion on ways of reducing choice complexity.

2.3.1 Register Array

The register array has two distinct parts, both of which are built from a basic dual-port static RAM cell: a 32×32 general purpose register set and working register set of two temporary registers and a constant (255) register. The temporary registers and constant are used chiefly for the character operations. The 32 general purpose registers provide considerable flexibility for the software designers.

Design of RAM cells has been a particularly knotty problem for other microprocessor projects. Although the Quarter Horse RAM cell was specially designed, it was based on a dual-ported cell that was produced from a VLSI circuit design generator developed at the University of Washington[1]. This experience illustrates two ways in which choice complexity can be reduced.

First, the structures like register arrays should be automatically produced using VLSI design generators just as PLAs are automatically generated now. The generator program is an encoding of the "experience" of the generator writer packaged in an extremely usable form. But not every architectural structure can be anticipated, so there may not be a design generator to solve a particular problem. Thus the second, and perhaps the most time-honored means of utilizing other designer's experience, is to *modify an existing* design. When the existing design is "close" to what is needed this is an extremely effective technique to reduced choice complexity. But when it is not "close" the existing design can possibly be a distraction preventing exploration of rich areas of the design space.

2.3.2 Arithmetic-Logic Unit

The ALU of the Quarter Horse is similar to that of the OM2 data path chip designed at CalTech [5]. The chief differences are that it was implemented in CMOS and like the Mosaic design [6], the "R function block" was replaced with an XOR gate.

The way in which choice complexity was controlled in the ALU was by utilizing the best features of designs produced by eight different designers. Students had been asked as a homework assignment in an introductory VLSI class to produce a CMOS version of the OM2 ALU. Eight completed designs were compared and the best parts from these were assembled into the Quarter Horse ALU.

Because the ALU could be produced quickly, it was possible to fix almost at

[1] The University of Washington/Northwest VLSI Consortium is presently engaged in a project to study and build design generators - flexible programs that produce circuits for standard architectural components. The RAM design generator was not mature enough to be used for the Quarter Horse directly. The only generators used were the pad frame generator and the PLA generator.

the beginning our 82μ data path pitch. This enabled subsequent design activities to progress with confidence that at least one characteristic of the design would not change.

2.3.3 Shifter

The barrel shifter, deemed to be a necessity because of our interest in data types requiring field extraction, was designed by beginning with published approaches [7,8]. The first design was generalized on a second pass to incorporate rotation. The use of a second layer metal CMOS process permitted a significant speed improvement due to reduced capacitance.

2.3.4 Instruction Register, PC and Memory Registers

The instruction register is a pair of registers to support instruction prefetch. These, plus the program counter, memory address and data registers, bus drivers, immediate and sign extension logic were all designed from scratch.

There is little that can be said about the management of choice complexity here except that we tried when possible to use library cells like flip-flops. In the end the designs were mostly original thinking.

2.4 The Control PLA

Having been impressed by the flexibility provided by the single control PLA of the Mosaic design [6] and a related scheme of the PP4 [3], we decided to adopt it rather than embrace the control precepts of other microprocessor chips. The use of the PLA would simplify the addition of new instructions at a later date and it would be the easiest way to control complex instructions requiring many microcycles. At the same time we knew that its performance would be the limiting factor in the speed of the microprocessor. This tradeoff between the costs of performance and the benefits of flexibility were a continual subject of discussion and experimentation. Although we have already realized many of the flexibility benefits, we have not yet determined what the total cost will be. But let us explain.

In terms of choice complexity, what is crucial about the use of the PLA as the single central controlling device of an architecture is that *it delays the binding time of critical architectural decisions.* A principle of computing is to delay binding decisions for as long as possible simply to retain flexibility, and because a PLA is programmable — a complete revision can be produced in an hour — changes can be made trivially right up to the end. The importance of the delayed binding principle for microprocessor architecture design was dramatically illustrated to us when three days before completion of the chip we had to change the general register read protocol! Moreover, as the simulations provided us with information about the performance of the architecture's components, we could continually revise the microcontrol. The conclusion has to be that the architecture itself can aid in controlling choice complexity. But at what cost?

The performance of the PLA concerned us from the very start and we simulated "typical" PLAs to get estimates of performance before adopting the architectural strategy. (The actual, as opposed to the psychological, effect on our work was minimal since we were going to redesign anyhow and the PLA could then be implemented in faster, random logic if it turned out to be too slow.) We took an aggressive 50ns cycle time as a goal and spent a lot of effort trying to achieve it. We experimented with a variety of design styles, domino, NORA, pseudo nMOS, etc., in order to find high performance solutions. We also enumerated a variety of ways in which a slow PLA could be made faster, but none was actually implemented. Most of this activity was to build confidence that the PLA-on-the-critical-path decision was correct. The Quarter Horse chip that was fabricated did not (by simulation evidence) meet the 50ns PLA requirements but by then the flexibility benefits alone justified the decision and slow performance was of less concern. (Measurements on the actual chip were not available at publication time, the simulated time for the PLA was 70ns-80ns.)

There was one other way in which the control PLA reduced choice complexity: By using the PLA for *all* timing, there was no interdependence among the architectural components on clock characteristics, which promoted more independence among the parts. If a single global clock had been used for each component, then there would have had to have been agreement among the designers on such things as the duration of each phase, even with a PLA control. With the PLA doing all of the timing, short duration activities like precharging can be done in one step while ALU computation can be given several cycles, i.e. the "logical clock" used by the component can be asserted for several "physical clock" periods. The lesson is to keep the clock out of the component designs.

2.5 Tools

The Quarter Horse chip was designed using Release 2.1 of the University of Washington/Northwest VLSI Consortium Design Tools [9]. Our chief tool for handling choice complexity was the RNL simulator of Chris Terman [10] which was applied to all component designs and to the entire chip. This tool, revised for CMOS and reasonably well calibrated to the MOSIS [11] processes, was an effective way to explore the design space quickly. For certain cells, e.g. RAM, we used SPICE simulation.

While designing the architecture, we built an interactive simulator to allow "register transfer level" simulation of the Quarter Horse. This program, which ultimately produced the microcode for the PLA, was an operational "document" that continually reflected the current state of the high level design decisions. Such tools enable one to try many architectural alternatives and they are worthy of greater exploitation in the future.

2.6 The Final Chip

The goal of designing a 32-bit microprocessor was adopted on January 14, 1985 and the Quarter Horse was queued for fabrication on April 15, 1985. It is composed of approximately 25,000 transistors and used the two layer metal 3μ p-well bulk CMOS process provided by MOSIS [11]. (The chip fabrication was not complete at publication time.) A complete description of the architecture can be found in The Architecture of the Quarter Horse Microprocessor [12].

3 Summary

Multiproject chips have made possible a reduction in the time required to fabricate a prototype chip to 1-3 months. The time required to design a prototype should be similarly brief. The Quarter Horse effort demonstrates that attention paid to choice complexity management permits substantial prototypes to be designed rapidly.

Although the Quarter Horse is not the last word, and there is much still to be studied about choice complexity management, it is useful to recapitulate the points cited above. First,

- **try a throw-away design as an intermediate goal.**

It is an effective way to acquire experience and streamline the effort. Second,

- **read the literature and avoid needless reinvention.**

No matter how inventive or creative a project is, it contains aspects that have been done before. Third, when possible

- **use design generators**

to avoid designing entirely, but if that is impossible, find something close and rework it.

- **Revise existing designs,**

and if several designers can be assigned to a critical part to do independent solutions,

- **merge the best of separate design efforts.**

The sixth rule may not always be applicable, but it was so important to the Quarter Horse, it is worth seeking cases to apply it:

- **Employ a flexible architecture that delays binding**

and plan to implement it with a generator tool such as a PLA. Finally,

- **build tools to aid exploration**

is a rule that will take the form of various simulators such as our architectural simulator.

ACKNOWLEDGMENTS:

We are indebted to the staff of the UW/NW VLSI Consortium for their help throughout the design: Without their direct technical and moral support, the Quarter Horse would not have made it to the finish line.

References:

1. Nelson F. Goncalves and Hugo J. DeMan, *Nora: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures*, IEEE Journal of Solid-State Circuits, SC-18(3): 261-266, (June, 1983).

2. Manaolis G. H. Katevenis, *Reduced Instruction Set Computer Architectures for VLSI*, Ph. D. Thesis, Univ. of California at Berkeley, October 1983.

3. W. D. Moeller and G. Sandweg, *The Peripheral Processor PP4: A Highly Regular VLSI Processor*, Proceedings of 11th International Symposium on Computer Architecture, pp. 312-318 (June 1984).

4. J. Hennessey, N. Jouppi, S. Przybylski, C. Rowen and T. Gross, *Design of a High Performance VLSI Processor*, Proceedings of 3rd CalTech Conference on VLSI, California Institute of Technology, Pasadena, California, pp. 33-54, March 1983.

5. Carver Mead and Lynn Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980, Chapter 5.

6. Christopher Lutz, *Design of The Mosaic Processor*, Masters Thesis, California Institute of Technology, May, 1984.

7. Robert W. Sherburne Jr., Manolis G. H. Katevenis, David A. Patterson, and Carlo H. Séquin, *Datapath Design For RISC*, Proc. of 1982 Conf. on Advanced Research in VLSI, MIT pp. 53-62.

8. John A. Bayliss, Stephen R. Colley, Roy H. Kravitz, Gary A. McCormic, William R. Richardson, Doran K. Wilde, Leon L. Wittmer, *The Instruction Decoding Unit for the VLSI 432 General Data Processor*, IEEE Journal of Solid-State Circuits, Vol. SC-16(5): pp. 531-536, (October 1984).

9. UW/NW VLSI Consortium, *Design Tools Release 2.1*, University of Washington, 1984.

10. Christopher J. Terman, *User's Guide to NET, PRESIM, and RNL/NL*, Technical Report, Massachusetts Institute of Technology, (September 1982).

11. Danny Cohen and George Lewicki, *MOSIS – The ARPA Silicon Broker*, Proceedings of the CalTech Conference on VLSI, California Institute of Technology, Jan. 1981.

12. S. Ho, B. Jinks, T. Knight, J. Schaad, L. Snyder, A. Tyagi, and C. Yang, *The Architecture of the Quarter Horse Microprocessor*, Technical Report, University of Washington, 1985.

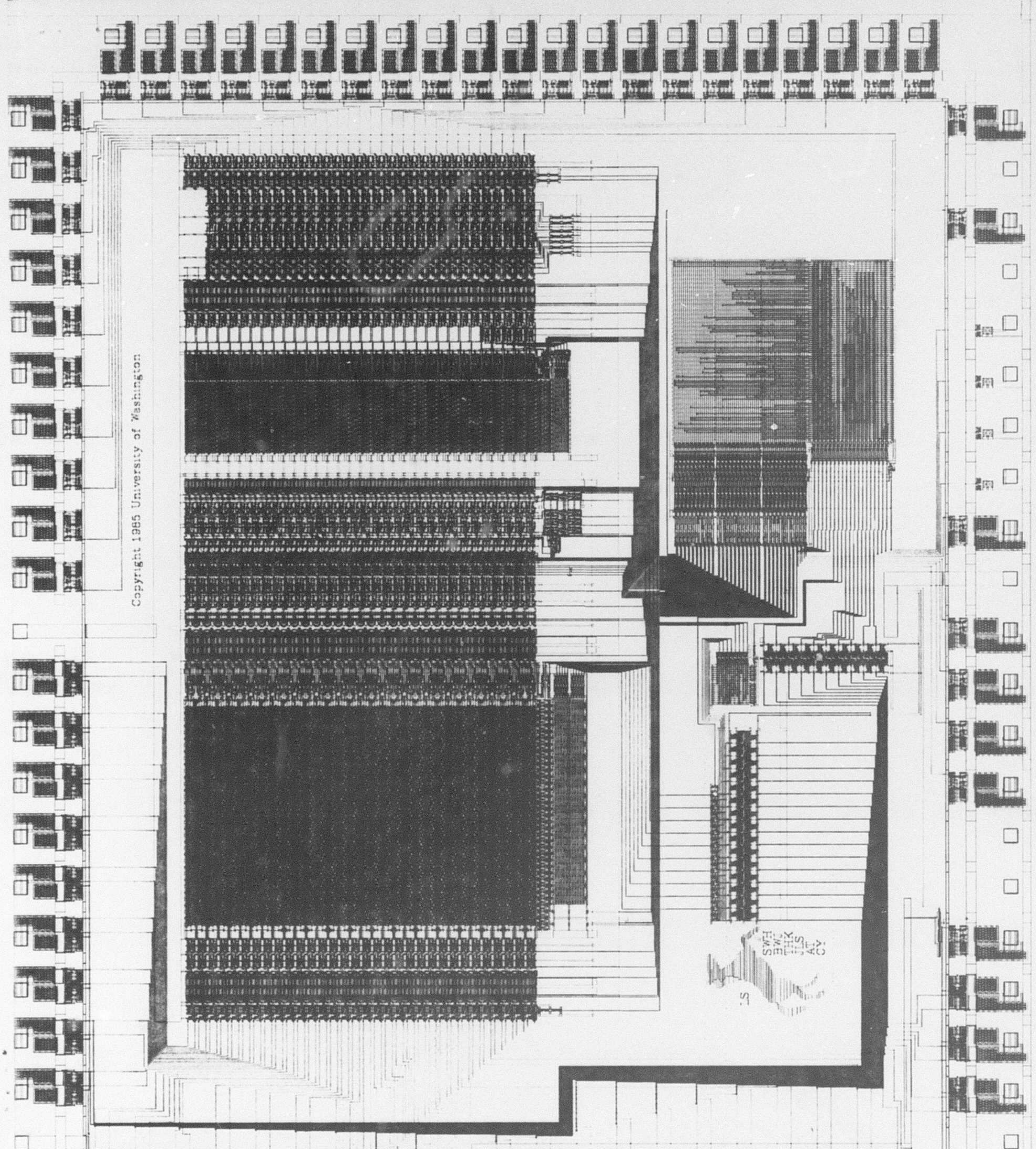

Figure 1. Quarter Horse check plot.

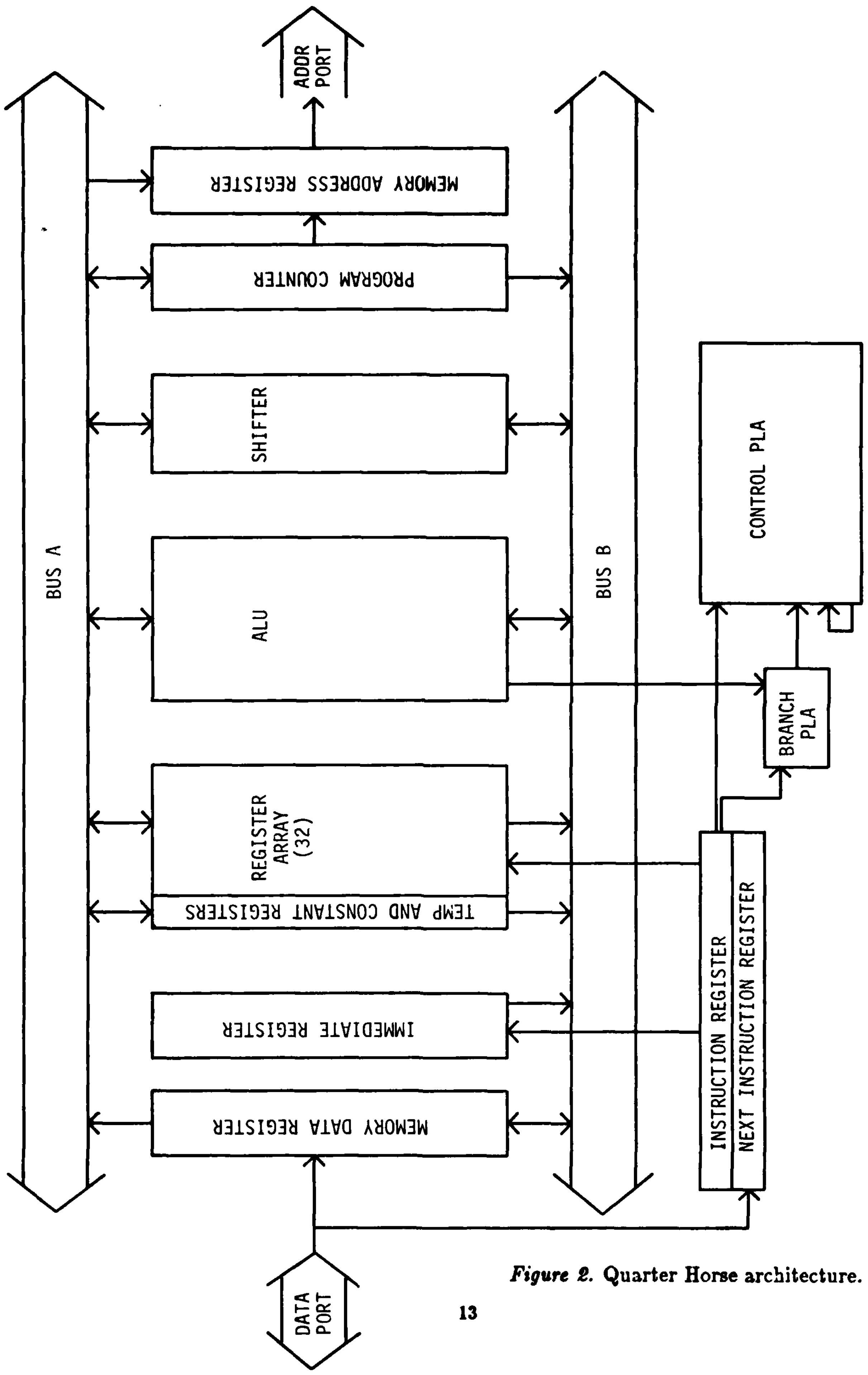

Figure 2. Quarter Horse architecture.